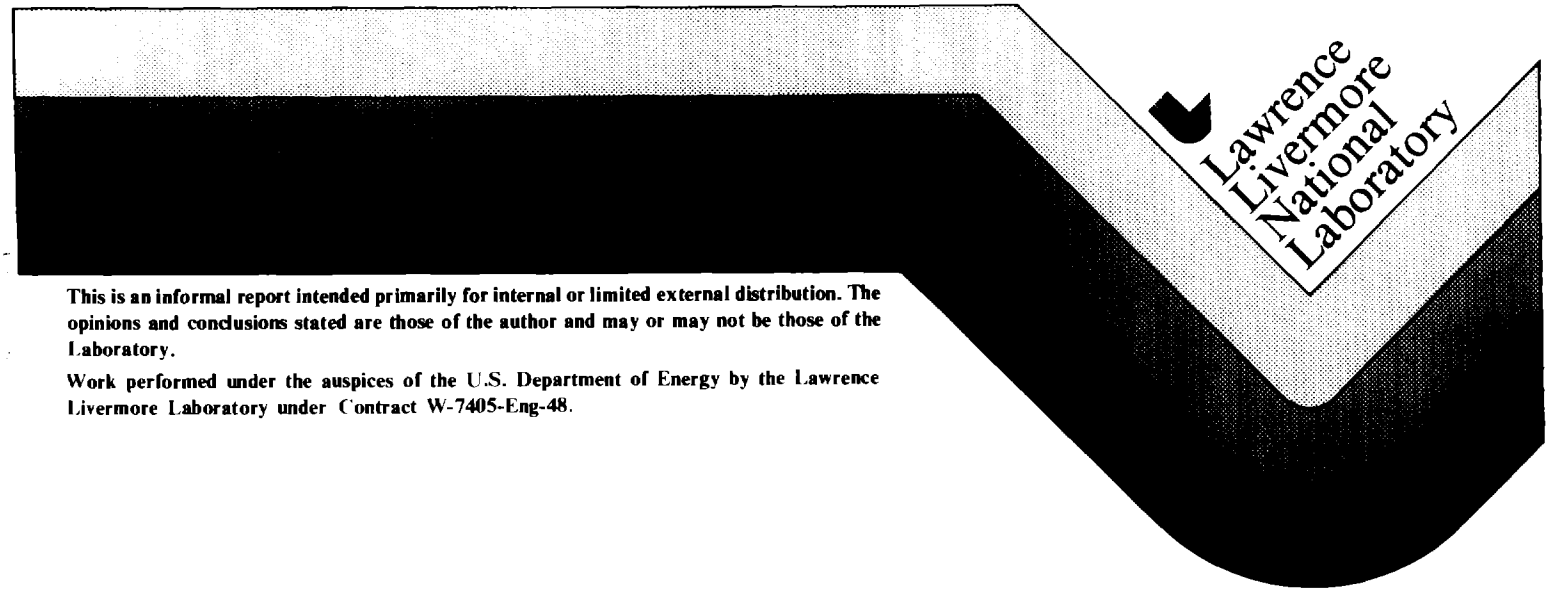
UCID-20478

Installation of the PADL-2 Solid Modeler on the CRAY-1

J. R. Kalibjian

August 1985



Work performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore Laboratory under Contract W-7405-Eng-48.

Printed in the United States of America
Available from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road
Springfield VA 22161
Price: Printed Copy $; Microfiche $4.50

Page Range	Domestic Price	Page Range	Domestic Price
001-025	$ 7.00	326-350	$ 26.50
026-050	8.50	351-375	28.00
051-075	10.00	376-400	29.50
076-100	11.50	401-426	31.00
101-125	13.00	427-450	32.50
126-150	14.50	451-475	34.00
151-175	16.00	476-500	35.50
176-200	17.50	501-525	37.00
201-225	19.00	526-550	38.50
226-250	20.50	551-575	40.00
251-275	22.00	576-600	41.50
276-300	23.50	601-up[1]	
301-325	25.00		

[1]Add 1.50 for each additional 25 page increment, or portion thereof from 601 pages up.

Installation of the PADL-2 Solid Modeler on the CRAY-1

Abstract

We describe in detail the method used to install the PADL-2 solid modeler on the CRAY-1 at Lawrence Livermore National Laboratory (LLNL), and document the solutions required to successfully implement two versions of this VAX native source code. Inadequate PADL-2 documentation and PADL-2's dependence on many non-standard FORTRAN features delayed bringing up Version 1.0. Additional problems resulted from little known quirks in LTSS system routines such as 8-bit character data losses (in CIVIC) on byte data transfers to subroutines and binary incompatibility between CIVIC and CFT logical variables and syntax conflicts with PRECOMP.

We installed the second version, PADL-2/1.2, with much less difficulty because the code was cleaner and most implementation problems had already been identified. Yet, other problems were discovered such as CFT compiler differences from standard FORTRAN-77 and lack of LTSS support libraries for list directed and file I/O.

Finally, modifications to the original Lexidata and Tektronix graphic interface routines let us generate (1) color high-resolution, shaded pictures on the laboratory's Dicomed film recorder, (2) low-resolution black-and-white dithered pictures of shaded objects displayable on TMDS and RJET, and (3) wireframe pictures of parts also displayable on both TMDS and RJET.

Introduction

Solid modelers[1] produce a complete mathematical description of a user defined object. The primary application goal is to automate the analysis and manufacture of mechanical parts. Solid modeling research[2] began at LLNL approximately four years ago. The basic goal of the project was to advance algorithm research in two areas; specifically, in extending the geometric coverage and graphics capabilities of solid modeling systems.

Because we desired to implement the algorithms developed in our research on actual solid modeling systems, we wanted to acquire modelers that could be installed on the Livermore Time Sharing Computer System (LTSS). When the research effort began, there were only two solid modeler source codes available to research institutions at a reasonable cost. They were TIPS-1,[3] a CSG-like (constructive solid geometry) modeler produced at Hokaido University that was available through CAM-I, and PADL-2,[4] a hybrid modeler produced by the Production Automation Project at the University of Rochester. We chose to implement both TIPS-1 and PADL-2. TIPS-1 was brought up in 1982 and has since undergone many changes. PADL-2 Version 1.0 was brought up in July 1984; Version 1.2 was working by February 1985.

Strategy for Installation

Versions 1.0 and 1.2 of PADL-2 were native to the VAX computing environment. Installing the code on a CRAY-1 revealed many little known facets of the LTSS FORTRAN environment and helped identify important differences between VAX,[5] CRAY,[6] and Livermore[7] FORTRAN. A three-step process was employed to get PADL-2/1.0 working on the CRAY-1. Step one encompassed moving the PADL-2 source code to the CRAY and modifying it for use in the LTSS environment. Step two was static debugging. In this phase, device dependencies were first eliminated;

then all the PADL-2 sources were pre-processed from the PADL-2 source language (FLECS) to FORTRAN and then compiled. Finally, step three was dynamic (run-time) debugging on the PADL-2 system. This three-step process proved to be very effective, and it was also used to bring up Version 1.2.

Moving PADL-2 Sources to the CRAY

Moving the PADL-2 sources to the CRAY was time consuming since PADL-2 comprises about 600 subroutines, totalling approximately 50,000 lines of code. The University of Rochester delivered the source files on a tape written with their VAX in a 24 × 80 blocked ASCII format. It was read onto the CRAY using the OCTOPUS utility RWFILES.[8] Several intermediate format modifications were then made to convert the files to CRAY format files. After the CRAY formatting was complete, the files were reorganized and saved. On the VAX, logically related files were grouped together via a subdirectory structure. On the CRAY, this was done by placing related source files in the same "LIB"[9] library and saving the library under one root directory.[10]

Modifying Sources for the LTSS Environment

Before the PADL-2 routines could be compiled on the CRAY, we had to modify them for use in their new programming environment. One of the first problems we faced was an illegal character introduced into the PADL-2 source files by Rochester's use of the TAB function on their terminals.

Illegal Character Introduced by Tab Function

At Rochester, programmers used the TAB function key to position their cursor at column seven in preparation for typing a line of FORTRAN code. Unfortunately, when a file containing such a code is read from tape with RWFILES or TAPECOPY[11] as a 24 × 80 blocked ASCII file, the TAB maps as an illegal ASCII character that is replaced with a question mark (ASCII 63). Thus, it was necessary to replace question marks in column one with at least seven spaces in all files that made use of the TAB function.

Include Directives

Another problem dealt with VAX INCLUDE directives in the PADL-2 source files. Most PADL-2 routines made use of common block files. For example, if a VAX FORTRAN routine A.FOR uses common block file B.COM, then A.FOR must declare INCLUDE B.COM. To do this in LTSS, B.COM must first be made a cliche file[12]; then A.FOR must declare USE B.COM. All files were checked and corrected for this problem with a TRIX/COSMOS controllee.[13,14]

Eight-Bit Character Data

The first two problems encountered with PADL-2 files were system dependent, but the last major problem in the modification for LTSS was a FORTRAN implementation issue—the handling of 8-bit character data. To allow for maximum transportability, Rochester programmers placed character data in 8-bit LOGICAL*1 arrays. Further, character data to be archived in the PADL-2 data base was placed in a LOGICAL*1 array equivalenced to the PADL-2 main storage area. This scheme was quite efficient and appropriate in the VAX environment; yet, it was unworkable on the CRAY because LOGICAL*1 variables are 64 bits long, and because variables of different type (other than real and integer) could not be equivalenced using the CRAY CFT compiler.

A method for implementing character handling using LRLTRAN, which allows mixed mode equivalencing, had to be devised. Before this could be done, all routines that used LOGICAL*1 variables and the equivalencing technique had to be located. This was difficult because Rochester did not clearly indicate which routines used the LOGICAL*1 and the equivalencing facilities. Eventually a TRIX/COSMOS controllee found all "offending" routines, and the LOGICAL*1 variables were simply declared as BYTE <*varname*>(8) variables. This declaration did not eliminate all the difficulties; we discovered that in LRLTRAN, 8-bit arrays passed between subroutines have their data right shifted, and hence lost. To resolve this, all 8-bit character arrays passed between subroutines were equivalenced to 64-bit integer arrays. The 64-bit integer arrays were then passed as parameters between the subroutines.

Graphics Interfaces

In addition to these VAX FORTRAN and system dependencies in the PADL-2 software, there were still other dependencies in the code related to graphics devices. Fortunately, these dependencies were well documented and easily fixed.

At Rochester, a Lexidata 3400 raster device displayed high-resolution shaded images and a Tektronix terminal was used for most line

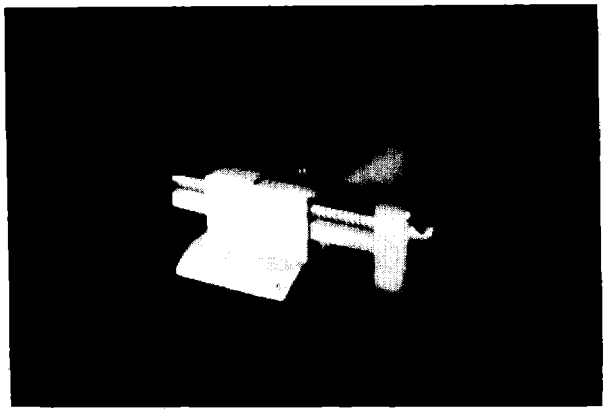


Figure 1. Color-shaded Dicomed picture of a part generated by PADL-2; CRAY time = 14.6 s.

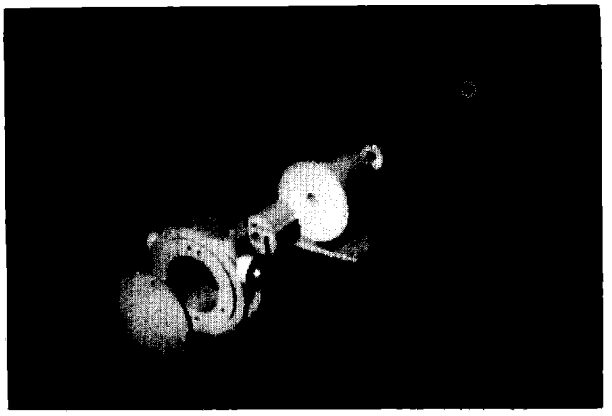


Figure 2. Color-shaded Dicomed picture of a part generated by PADL-2; CRAY time = 16.0 s.

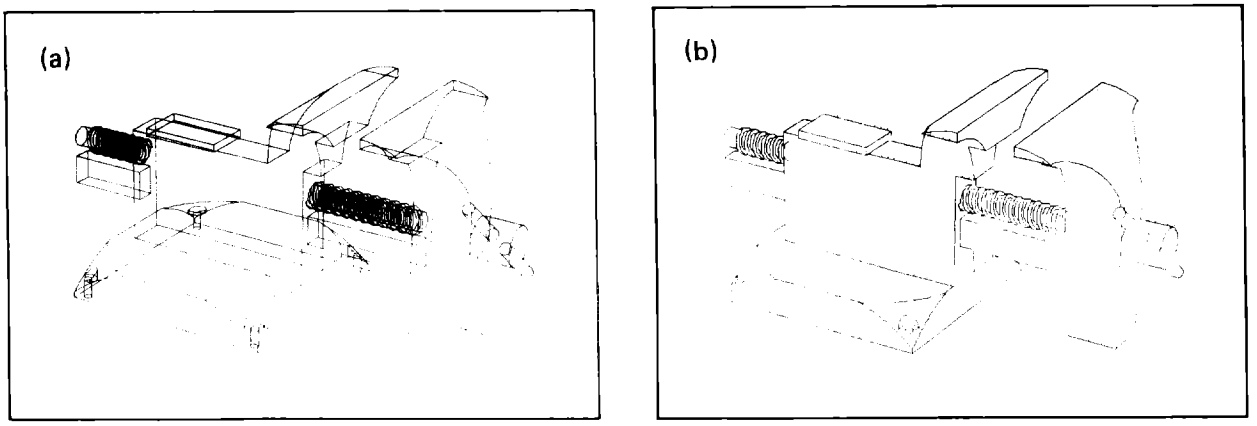


Figure 3. Wireframe image of PADL-2 part in Fig 1(a)showing hidden lines, CRAY time = 0.6 s and (b) with hidden lines removed, CRAY time = 32.7 s.

drawings. We modified the graphics interface routines for the Lexidata to produce Dicomed film-recorder (red, green, and blue) absolute files and a dither file. With the Dicomed files, color shaded pictures of objects could be obtained in several formats: 4- × 5-in. polaroids, slides (Figs. 1 and 2), and 8- 1/2- × 11-in. glossies. The modifications made to the Tektronix interface routines allow us to display wireframe images on TMDS or RJET as shown in Figs. 3(a), 3(b), 4(a), and 4(b) using the graphics library PLOTLIB.[15] Finally, the dither file contains data used to display low-resolution, black-and-white shaded images on TMDS, or on RJET.

Pre-Processing

After we modified the PADL-2 sources for the LTSS environment, they had to be pre-processed from FLECS[16] to FORTRAN-66[17] and then compiled. The FLECS pre-processor also had to be modified to run on OCTOPUS. The FLECS code had two VAX dependencies: (1) the 32-bit word size of the VAX and (2) a few, non-standard, VAX FORTRAN features (e.g., byte variable allocation). Once we fixed these two dependencies, and solved a problem with an unlabeled common block, the FLECS pre-processor functioned well.

Compilation

After pre-processing, the PADL-2 source files were compiled with either CIVIC or CFT (this caused difficulties later), and placed in BUILD[18] binary libraries. Because there were so many files, a COSMOS controller managed the pre-processing and compilation of the PADL-2 sources. Compilation uncovered a number of errors in the code, including misdeclared variables,

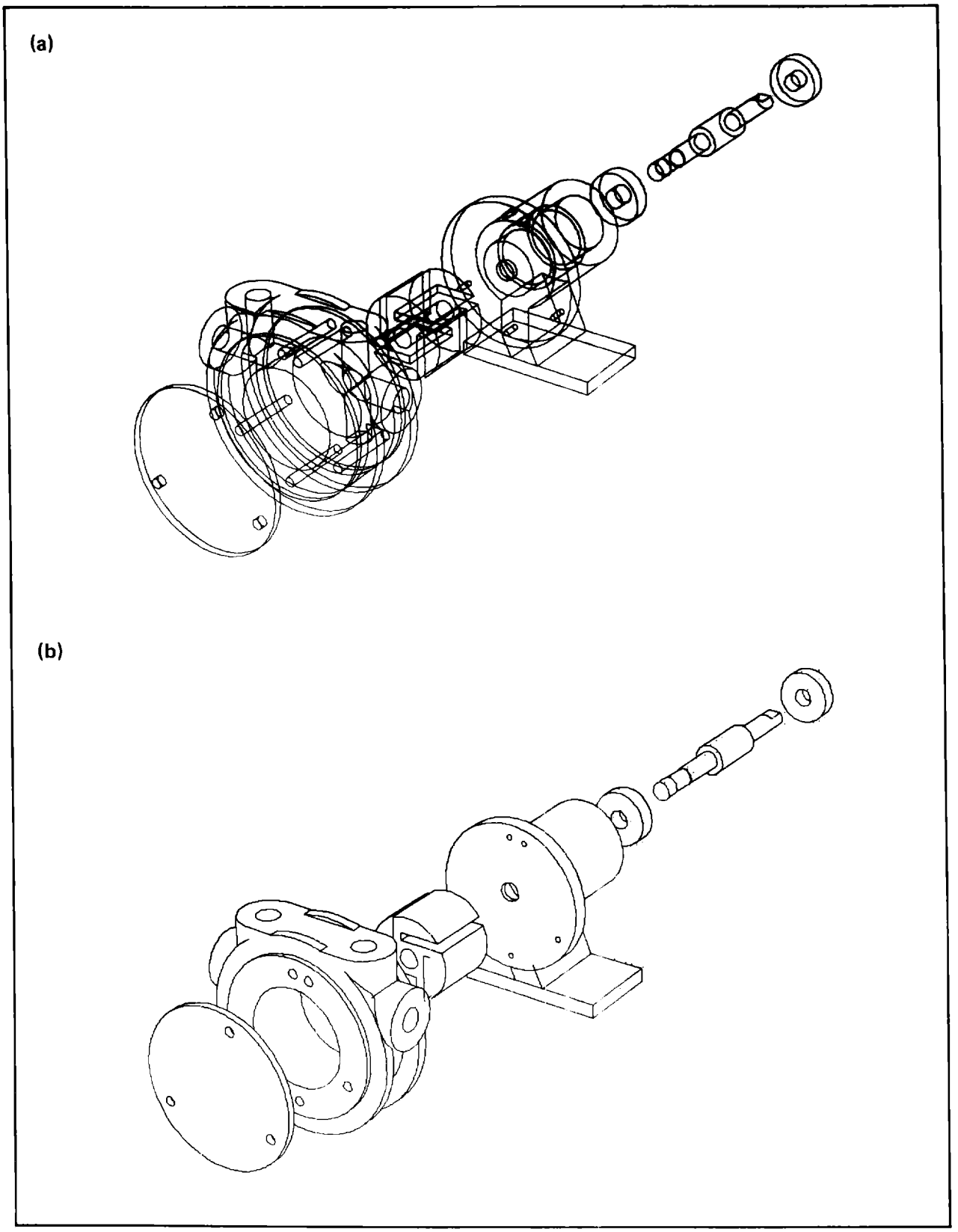


Figure 4. Wireframe image of the PADL-2 part in Fig. 2(a) showing hidden lines, CRAY time = 11.5 and (b) with hidden lines removed, CRAY time = 26.6 s.

mislabeled subroutines or functions, missing return statements in subroutines, and lines over 80 characters long.

Run-Time Debugging

After all the PADL-2 sources were compiled, they were loaded using LDR[19] and a PADL-2 controllee was generated (size = 0.83M CRAY words). Of course, PADL-2 did not work the first time, so run-time debugging with the OCTOPUS utility DDT[20] was the next step. Our basic approach was to run a VAX version of PADL-2, installed on the Non-Destructive Evaluation (NDE) VAX at LLNL, in parallel with the CRAY-1 version of PADL-2/1.0. We set breakpoints in important areas of the code, and examined key variables at each breakpoint. Different variable values for the VAX and CRAY versions at the same breakpoint indicated that something was amiss. Yet this did not end the battle; the hard part was determining what caused the discrepancies between two versions with identical source codes. Finally, after some time, we identified three problems that prevented successful execution of PADL-2 on the CRAY-1.

The first problem was a binary incompatibility between CFT compiled code and CIVIC compiled code. Although it was not widely advertised, CFT and CIVIC binary codes were not completely compatible. This was because in CIVIC, logical TRUE is represented with the integer one, while FALSE is represented with the integer zero. In CFT, logical TRUE is any value less than zero (usually negative one), while FALSE is any value greater than or equal to zero (usually zero). Thus, if a routine compiled by CFT passed a logical value to a routine compiled by CIVIC, or vice-versa, problems would occur. This was indeed happening in some parts of the CRAY-1 version of PADL-2. Now, recall that some parts of PADL-2 were compiled with CIVIC and others with CFT. This was done because each compiler has features that were more desirable to use on certain parts of the PADL-2 code. However, because of the inconsistent logical values between CFT and CIVIC, only one compiler could be used. We chose the CIVIC compiler, and re-compiled all CFT-compiled routines.

The second problem dealt with block data directives. Unlike VAX FORTRAN, CIVIC will not automatically load data declared below a block data statement. To get around this difficulty, we converted the block data segments into subroutines and declared them as external in the PADL-2 main driver program.

The third and final problem was in the initialization of an integer function. In VAX FORTRAN, integer functions have a default value of zero if no value is assigned to the function variable in the function body. Unfortunately, this was not the case in LRLTRAN. We identified one function in the PADL-2 code that was not working properly because of this problem. Once this problem was corrected, Version 1.0 of PADL-2 was up and running on the CRAY-1.

Version 1.2

The experience we gained from bringing up Version 1.0 helped expedite the installation of Version 1.2 on the CRAY. This second version was much cleaner than Version 1.0. For instance, the University of Rochester eliminated all character handling problems by placing character data in FORTRAN-77 CHARACTER*1 arrays. Yet Version 1.2 contained other interesting problems, primarily those caused by the new FLECS pre-processor shipped with the second version. This particular FLECS pre-processor converted FLECS code into FORTRAN-77. Because CIVIC was not compatible with FORTRAN-77, we had to use the CFT compiler to compile the pre-processed modules. However, the same three-step implementation process outlined for Version 1.0 was employed.

Moving Version 1.2 Files to the CRAY

Moving the Version 1.2 sources from tape to the CRAY proved to be slightly more difficult than with Version 1.0. Rochester sent a VAX/VMS backup format tape containing the PADL-2 sources, instead of a 24 x 80 blocked ASCII format tape. Thus, we had to reformat the tape for the utility routines TAPECOPY[20] or RWFILES. The files were read onto a VAX and rewritten to replace their variable length records with 80-column lines. Then the files were written back to

tape using the VAX COPY command and read onto the CRAY. To expedite file modification for LTSS, we wrote a pre-processing program to take care of the INCLUDE and TAB problems described earlier.

Static Debugging of Version 1.2

After the files were modified, the FLECS/FORTRAN-77 pre-processor was installed on the CRAY. The FLECS pre-processor was written in FORTRAN-77, so it had to be compiled with CFT. However, it was not clear just how much of the standard FORTRAN-77[21] was in the CFT compiler because many CFT FORTRAN-77 features were not well documented. We discovered that the newest CFT compiler had most of the FORTRAN-77 features, with two exceptions: (1) CHARACTER* variables could not be declared with a length greater than 504 characters and (2) integer and real variables could not be written to character strings. The FLECS pre-processor took advantage of these standard FORTRAN-77 features, and we wrote routines in the CFT subset of FORTRAN-77 to implement them. After doing this, the FLECS pre-processor was easily brought up on the CRAY.

Once the FLECS pre-processor was working, pre-processing and compiling PADL-2 sources with a COSMOS controllee could begin. We found two problem areas in the Version 1.2 source files: specifically, list-directed I/O and the IMPLICIT NONE directive.

The CFT compiler could generate externals to handle list-directed I/O and file I/O. Unfortunately, no system library in LTSS could satisfy the generated externals. Therefore, we created a pre-processing program to translate FORTRAN-77 list-directed I/O statements into standard FORTRAN-66 WRITE directives. Since file I/O was done in only a few PADL-2 routines, those calls were manually replaced with equivalent FORTLIB[22] calls.

While the CFT compiler technically conformed to the FORTRAN-77 (and VAX FORTRAN) standard with regard to I/O, this was not the case with the IMPLICIT NONE directive. In the VAX FORTRAN, IMPLICIT NONE essentially requires that all variables and functions be explicitly declared in a subroutine module. Yet in the CFT interpretation, subroutines must also be declared via the EXTERNAL statement. To alleviate this problem, the list-directed I/O processor was modified to check for, and delete, the IMPLICIT NONE statement in PADL-2/1.2 source files.

Once all the source files were compiled, we loaded them using LDR, generating a controllee 0.85M CRAY words long. Surprisingly, the controllee worked the first time and no run-time debugging was necessary. This fortunate occurrence was probably due to two factors: (1) the cleaner version of PADL-2 that Rochester developed to conform to the FORTRAN-77 standards, and (2) identification of most potential pitfalls during our implementation of Version 1.0.

Conclusions

To summarize, implementing Versions 1.0 and 1.2 of PADL-2 on the CRAY revealed the following facts:

- While PADL-2 is a landmark in solid modeling research and development, it is poorly documented. This prevented us from bringing it up quickly on the CRAY-1.
- LRLTRAN is ill equipped to deal with 8-bit character data efficiently because it right shifts the byte data passed between subroutines.
- Integer functions in LRLTRAN are not automatically set to zero upon entry.
- There is a binary incompatibility between CIVIC and CFT logical variables.
- Most FORTRAN-77 character handling features are implemented in the new CFT compiler and the following intrinsic character handling functions are available: INDEX, LEN, CHAR, ICHAR, LGE, LGT, LLE, and LLT. However, CHARACTER* variables cannot have unlimited length, and integer and real values cannot be written to strings.
- The CFT compiler can generate externals for list-directed I/O and file I/O, but no LTSS support library exists that satisfies these externals.
- The IMPLICIT NONE statement in VAX FORTRAN has a different effect than the IMPLICIT NONE statement in CRAY FORTRAN.

- In order to write files from a VAX to tape so they can subsequently be read by RWFILES or TAPECOPY on OCTOPUS, observe the following:
  a) Make sure the files do not have variable length records.
  b) Use the VAX COPY command to write the files to tape.
  c) Once read off tape, the files must undergo the following RWFILES conversions: A8A6., CPLR.80, and A6CRAY.
- Data statements involving the < or the > character in a cliche file cannot be handled correctly by PRECOMP. This is because the < and the > characters have a special meaning in PRECOMP syntax.

- *The following bug has been fixed, and it is described for documentation purposes only.* Assume module A (compiled with CFT) performs I/O using FORTRAN-77 CHARACTER* variables. Later, if module B (compiled with CIVIC) is called and tries to perform character I/O, by trying to read characters into an integer variable for instance, it will not succeed. Instead, only one character will be read into the integer word. A consultant discovered that when FORTRAN-77 I/O was being done, an internal variable in an I/O routine was set to one, indicating that only one character could be stored in an 8-bit word. Unfortunately, this variable was not re-initialized when I/O in a CIVIC procedure was initiated.

Acknowledgments

Special thanks to R. P. Yaffee, T. C. Michels, J. R. Matthews, D. E. Sacket, T. G. Allison, R. Crowe, D. L. Vickers, L. E. Taylor, M. D. Blair, M. K. Kong, and G. W. Laguna.

References

1. J. R. Kalibjian, *Solid Modeling: Foundation for Manufacturing Automation*, Lawrence Livermore National Laboratory, Livermore, CA, UCRL-53651 (1985).
2. J. R. Kalibjian, *Solid Modeling Research at Lawrence Livermore: 1982–1985*, Lawrence Livermore National Laboratory, Livermore, CA, UCRL-53652 (1985).
3. Norio Okino, Yukinori Kakazu, and Hiroshi Kubo, "Theories for Graphics Processors in TIPS-1," *Computers and Graphics* **7**, 3–4, pp. 243–258, 1983.
4. Christopher M. Brown, "PADL-2: A Technical Summary," *IEEE Computer Graphics and Applications*, **2**(2), pp. 69–84, March, 1982.
5. Digital Equipment Corporation, *VAX-11 FORTRAN Reference Manual*, April 1982.
6. Cray Research Inc., *CFT: The CRAY FORTRAN Compiler*, Lawrence Livermore National Laboratory, Livermore, CA, LCSD-304, February 13, 1981.
7. William S. Derby, John T. Engle, and Jeannie T. Martin, *LRLTRAN Language Used with the CHAT and CIVIC Compilers*, Lawrence Livermore National Laboratory, Livermore, CA, LCSD-302, June 1, 1982.
8. Lawrence D. Sears, *RWFILES User's Manual*, Lawrence Livermore National Laboratory, Livermore, CA, LCSD-1175.
9. Rich D. Bellas, *LIB*, Lawrence Livermore National Laboratory, Livermore, CA, LCSD-1548, October 2, 1981.
10. Rich Bellas and Neale Smith, *XPORT Reference Manual*, Lawrence Livermore National Laboratory, Livermore, CA, LCSD-1196, December 1, 1981.
11. Jim Minton and Jeffrey C. Huskamp, *TAPECOPY*, Lawrence Livermore National Laboratory, Livermore, CA, UR-403, October 10, 1977.
12. Dennis Johnson, *PRECOMP*, Lawrence Livermore National Laboratory, Livermore, CA, LCSD-328, September 28, 1979.
13. Gary Long, *TRIXGL*, Lawrence Livermore National Laboratory, Livermore, CA, LCSD-832, February 2, 1984.
14. Clement Luk and Bruce Kelly, *COSMOS*, Lawrence Livermore National Laboratory, Livermore, CA, LCSD-508, May 16, 1984.
15. Mark Blair, Jim Kohn, Eric Mueller, and Lynd Stringer, *PLOTLIB User's Manual*, Lawrence Livermore National Laboratory, Livermore, CA, LCSD-439, April 1983.
16. Delshler Armstrong, *FLECS/77 User's Manual*, University of Rochester (1978), pp. 1–29.
17. American National Standards Institute, *Programming Language FORTRAN ANSI X3.9-1966* (American National Standards Institute, NY, 1966).
18. Larry Berdahl, *BUILD*, Lawrence Livermore National Laboratory, Livermore, CA, LCSD-1512, January 20, 1981.
19. Rick Johnson, *LDR*, Lawrence Livermore National Laboratory, Livermore, CA, LCSD-344, October 23, 1984.
20. Dave Seberger, *DDT*, Lawrence Livermore National Laboratory, Livermore, CA, LCSD-1620, November 24, 1981.
21. American National Standards Institute, *Programming Language FORTRAN ANSI X3.9-1978* (American National Standards Institute, NY, 1978).
22. Barbara Atkinson, *FORTLIB—A Standard FORTRAN Library for the CDC 7600 and the CRAY-1*, Lawrence Livermore National Laboratory, Livermore, CA, LCSD-406, January 14, 1982.

PMB/mlc